CS 678 Machine Learning

Project Assignment #2

David Qorashi
Ehsan Valizadeh

In this project we tried to implement a Naive Bayesian Classifier. We got surprised by the accuracy of the results we obtained. The project is implemented in Ruby. As you probably know, the language itself is not designed for heavily computational tasks.

We encountered some problems due to this fact; we will address them at the following.
Our project is consists of two files: file_utility.rb and naive_bayes_classifier.rb.
First file tries to provide interfaces for reading training data from the file system. It goes line by line and at the end will extract the list of classes, vocabularies and records. We will use these three variables in the training section.

Spec

We executed our code on a new Dell PowerEdge T420 server (the MASL new server) with following specification:
Intel® Xeon® E5-2430 2.20GHz, 15M Cache, 7.2GT/s QPI, Turbo, 6Core, 95W, Max Mem 1333MHz

Approach

We have three primary methods in our naive_bayes_classifier.rb: train, classify and classify_whole_file. I will describe them one by one.

Train

The train(path) method is the heart of our Naive Bayes Algorithm.
First, It will read from an input file on the filesystem and with utilizing the codes written in file_utility.rb, it tries to extract the records, classes and words.

Then, it goes through the learning algorithm provided in the project description and at the end it

will return classes, vocabulary, probability and word_probability variables.

- **Classes**: An array consisting the whole classes our training data belonged to.
- **Vocabulary**: An array consisting of a list of the words we have in our training data.
- **Probability**: A ruby hash. It take a class (document type) as the key and for the value it preserves the **probability** estimate of that particular class.
- **word_probability**: A ruby hash. It shows the probability of word occurrence for particular document class.

Classify

After the training is complete it's the time for classification. Nothing vague goes on here. We're simply multiplying the word_proabability - we obtained it from our training part for every word in every class - by the word probability of other words in a document. This document is the one we need to classify.
At the end, we're returning the class with maximum probability. This is the oracle that our classification function made.

classifiy_whole_file

This file basically is reading from a file on the filesystem and tries to run classify method on every single document. It's running some analytics on the data too.

Interesting Problems

First, we had some performance problems. The training code was taking a long time to complete. (For the provided sample data, 3 hours)

We tried to rewrite the training part. There were some unnecessary loops. For example we were doing something in a loop and again we were running exactly the same loop to do another thing. We just merged these tasks and put them in one single loop. With these small improvements, we made the training part for forumTraining.data to run in 12 seconds.

We rewrote the counting part of training code in C language. We created this piece of code as a library method and used it directly in our Ruby program. With doing this, the total timing of the training data decreased to 1.3 seconds (average).
For the sake of clarity, I just brought the Ruby code (12 seconds version) in this document.

For the classification part (reading records from a file and then predicting the type of the documents), we did the following scenario.
For the sample forumTest file, the classifier took 62 minutes and 48 seconds to complete at first.

For the sample forumTest-stemmed file, the classifier took 21 minutes and 34 seconds to complete.
It's not very fast. We tried to optimize classification code first and then we tried to parallelize some possible parts. We tried to achieve more speedups.

First, we tried to specify which part of classify method is creating this kind of slowness.

```
def classify(classes, vocabulary, class_probablity, word_probability, document)
  words = document.split(' ').select {|w| vocabulary.include? w }
  products = {}

  classes.each do |c|
    product = BigDecimal(1)
    product *= class_probablity[c]
    words.each do |word|
      product = product * word_probability[c][word]
    end
    products[c.to_sym] = product
  end
  key = products.max_by{|k,v| v}[0]
end
```

As you see it's not a lengthy code and tracing the problem wasn't a tedious task. We noticed that the BigDecimal class is creating this lingering in the system.
BigDecimal provides arbitrary-precision floating point decimal arithmetic. We use it in Ruby for mathematical computation on very big or small numbers. We tried to change it to a typical float data type (in Ruby there is no double type as we have in language like C). The code run a lot faster with this change (12 minutes for forumTest data); although there was a problem. We lost a great percentage of correct oracles due to losing precision. The float data type in Ruby can preserve values greater than 10^{-323}. Apparently in the probabilities we had, there were some decimal numbers less than this value. When we were using the float datatype every number less than 10^{-323} converted to 0. So inevitably we kept using BigDecimal data type.

To achieve a better speedup, we used a ruby library called peach (parallel each) which breaks down a loop into several parts and run each part on a cpu core. As you know, MRI ruby (de facto standard) uses a mechanism called GIL (Global Interpreter Lock). Attaining real parallelism isn't possible using this concept. As a solution, we run the parallelized code in jruby and rubinius. Rubinius is an implementation of Ruby designed for concurrency. It is using native threads to run Ruby code on all the CPU cores.

Effectiveness of classifier

Training Set: Forum-Training.data
Test Set: Forum-Test.data

Number of total records in testing set	Number of correct oracles	Number of false oracles	Success Rate
7528	6041	1487	%80

We run an analytics on the false oracles. We compared the real category of a record and its corresponding oracle. In case they weren't equal (means that the oracle was wrong), we counted the number of every false prediction for that real_category:oracle pair and then sort them in a decreasing order.
For example as the first record, we have ['politics', :guns], 100.
It means that there were 100 politics document that mistakenly categorized into guns topic. The reason is obvious. There is a clear link between these two categories. The next one is xwindows documents which categorized under graphics part. A connection is shining here too.

Here is the result:

```
[[["politics", :guns], 100],
 [["xwindows", :graphics], 55],
 [["electronics", :pc], 50],
 [["mswindows", :pc], 48],
 [["religion", :atheism], 45],
 [["forsale", :pc], 41],
 [["xwindows", :mswindows], 34],
 [["religion", :christianity], 33],
 [["mideastpolitics", :atheism], 32],
 [["mac", :pc], 30],
 [["pc", :mac], 30],
 [["pc", :mswindows], 26],
 [["atheism", :religion], 24],
 [["graphics", :mac], 24],
 [["graphics", :mswindows], 22],
 [["medicine", :autos], 21],
 [["mideastpolitics", :politics], 19],
 [["forsale", :autos], 19],
 [["electronics", :mswindows], 19],
 [["forsale", :mac], 18],
 [["space", :politics], 18],
```

[["medicine", :atheism], 18],
[["graphics", :pc], 18],
[["medicine", :motorcycles], 16],
[["xwindows", :pc], 15],
[["mswindows", :graphics], 14],
[["pc", :electronics], 14],
[["motorcycles", :autos], 14],
[["religion", :guns], 13],
[["atheism", :christianity], 13],
[["electronics", :mac], 13],
[["mac", :mswindows], 13],
[["mswindows", :mac], 12],
[["christianity", :atheism], 12],
[["hockey", :baseball], 11],
[["space", :graphics], 11],
[["medicine", :electronics], 10],
[["space", :electronics], 10],
[["electronics", :motorcycles], 10],

Training Set: ForumTraining-stemmed.data
Testing Set: ForumTest-stemmed.data

Number of total records in testing set	Number of correct oracles	Number of false oracles	Success Rate
7528	6145	1383	%81

We run another analytics on the false oracles here. It's very similar to the unstemmed data mentioned above.

[[["politics", :guns], 103],
[["xwindows", :graphics], 59],
[["pc", :mac], 43],
[["religion", :atheism], 42],
[["xwindows", :mswindows], 34],
[["electronics", :pc], 34],
[["religion", :christianity], 33],

[["mswindows", :pc], 32],
[["forsale", :pc], 31],
[["pc", :mswindows], 29],
[["atheism", :religion], 28],
[["mac", :pc], 26],
[["mideastpolitics", :atheism], 24],
[["mswindows", :graphics], 21],
[["pc", :electronics], 20],
[["forsale", :mac], 20],
[["graphics", :mswindows], 18],
[["religion", :guns], 17],
[["mideastpolitics", :politics], 17],
[["christianity", :atheism], 17],
[["graphics", :mac], 16],
[["electronics", :mswindows], 16],
[["christianity", :religion], 15],
[["space", :graphics], 15],
[["forsale", :autos], 15],
[["electronics", :mac], 15],
[["electronics", :graphics], 13],
[["atheism", :christianity], 13],
[["mswindows", :mac], 13],
[["mac", :electronics], 12],
[["medicine", :autos], 12],
[["guns", :politics], 11],
[["space", :politics], 11],
[["medicine", :politics], 11],
[["medicine", :atheism], 11],
[["electronics", :cryptology], 10],
[["graphics", :xwindows], 10],
[["cryptology", :guns], 10],
[["graphics", :pc], 10],
[["mac", :mswindows], 10]

For the sake of brevity I skipped other records here. we will mention the complete copy for unstemmed data in Appendix A.

Further Investigation

New dataset

To assure that our classifier is working correctly, we created an small new classification problem by hand. We created a training set filled with some old memories with our friends. We classified each memory with a keyword (type) showing the main context of the memory. Among these were topics such as ‘love, funny, exam, food’. Our classifier succeeded to correctly predict the type of new memories we fed it by a success rate of %87.

Investigate the source of error

As we described before, the main source of error in our technique was the limitations on Ruby language itself. The BigDecimal type was slow and to be faster we switched to Float. Doing this, we lost a great deal of our precision and as a result our classifier stopped being very useful.

Here is effectiveness of our classifier with using Float instead of BigDecimal type.

Training Set: ForumTraining.data
Testing Set: ForumTest.data

Number of total records in testing set	Number of correct oracles	Number of false oracles	Success Rate
7528	1536	5992	%26

Training Set: ForumTraining-stemmed.data
Testing Set: ForumTest-stemmed.data

Number of total records in testing set	Number of correct oracles	Number of false oracles	Success Rate
7528	3107	4421	%41

Source Code:

#naive_bayes_classifier.rb

```
$LOAD_PATH << '.'
require 'file_utility'
require 'ostruct'
require 'bigdecimal'

private
def classify_documents_in_categories(classes, records)
  document = {}

  classes.each do |c|
    document[c] = []
  end

  records.each do |r|
    document[r.category] << r.words
  end

  document
end

def classify(classes, vocabulary, class_probablity, word_probability, document)
  words = document.split(' ').select {|w| vocabulary.include? w }
  products = {}

  classes.each do |c|
    product = BigDecimal(1)
    product *= class_probablity[c]
    words.each do |word|
      product = product * word_probability[c][word]
    end
    products[c.to_sym] = product
  end
  key = products.max_by{|k,v| v}[0]
end

def train(path)
  classes, vocabulary, records = FileUtility.read(path)

  probability = {}
  text = {}
  word_probability = {}
  count = {}

  document = classify_documents_in_categories(classes, records)


  classes.each do |current_class|
    probability[current_class] = document[current_class].size.to_f / records.size
    text[current_class] = document[current_class].join(' ')
```

```
    count[current_class] = {}
    vocabulary.each do |v|
      count[current_class][v] = 0
    end

    text[current_class].split(' ').each do |w|
      count[current_class][w] += 1 unless count[current_class][w].nil?
    end

    vocabulary.each do |word|
      word_probability[current_class] ||= {}
      word_probability[current_class][word] = (1 + count[current_class][word]).to_f /
(text[current_class].size + vocabulary.size)
    end
  end

  return classes, vocabulary, probability, word_probability
end

def classify_whole_file(path, classes, vocabulary, probability, word_probability)
  test_record = OpenStruct.new
  accurate_prediction_count = 0
  count = 0
  false_prediction_count = 0
  File.open(path, "r").each_line do |line|
    count += 1
    data = line.split(" ")
    test_record.real_category = data[0]
    test_record.content = data.drop(1).join(' ')
    oracle = classify(classes, vocabulary, probability, word_probability,
test_record.content)
    puts "oracle: #{oracle}, real: #{test_record.real_category}"
    if oracle == test_record.real_category.to_sym
      accurate_prediction_count += 1
    else
      false_prediction_count += 1
    end
  end

  puts "Total number of records: #{count}"
  puts "Number of correct predictions: #{accurate_prediction_count}"
  puts "Number of false predictions: #{false_prediction_count}"
end



##### Execution begins here #######

classes, vocabulary, probability, word_probability =
train('data/forumTraining-stemmed.data')
```

```
# To classify a whole datafile use this (datafile is a file containing a full document in
per line)
classify_whole_file('data/forumTest-stemmed.data', classes, vocabulary, probability,
word_probability)


# To classify only a document use classify method directly
# puts classify(classes, vocabulary, probability, word_probability, "contract for comput
law modul ask write comput contract for suppli comput softwar busi softwar written for
compani per chanc happen copi comput contract system mail comput contract contain sensit
materi can assur that will remain confidenti and will not pass view hbl graham wilson lsg
cck coventri law iii coventri univers")
```

```
#file_utility.rb
require 'set'
require 'ostruct'

module FileUtility

  def FileUtility.read(path)
    vocabulary = Set.new
    classes = Set.new
    records = []

    File.open(path, "r").each_line do |line|
      line_array = line.split(' ')
      record = OpenStruct.new
      record.category = line_array[0]
      record.words = line_array.drop(1)

      records << record

      record.words.each { |w| vocabulary.add(w) }
      classes.add(record.category)
    end

    return classes.to_a, vocabulary.to_a, records
  end

end
```

Appendix A:

[[["politics", :guns], 100],

[["xwindows", :graphics], 55],
[["electronics", :pc], 50],
[["mswindows", :pc], 48],
[["religion", :atheism], 45],
[["forsale", :pc], 41],
[["xwindows", :mswindows], 34],
[["religion", :christianity], 33],
[["mideastpolitics", :atheism], 32],
[["mac", :pc], 30],
[["pc", :mac], 30],
[["pc", :mswindows], 26],
[["atheism", :religion], 24],
[["graphics", :mac], 24],
[["graphics", :mswindows], 22],
[["medicine", :autos], 21],
[["mideastpolitics", :politics], 19],
[["forsale", :autos], 19],
[["electronics", :mswindows], 19],
[["forsale", :mac], 18],
[["space", :politics], 18],
[["medicine", :atheism], 18],
[["graphics", :pc], 18],
[["medicine", :motorcycles], 16],
[["xwindows", :pc], 15],
[["mswindows", :graphics], 14],
[["pc", :electronics], 14],
[["motorcycles", :autos], 14],
[["religion", :guns], 13],
[["atheism", :christianity], 13],
[["electronics", :mac], 13],
[["mac", :mswindows], 13],
[["mswindows", :mac], 12],
[["christianity", :atheism], 12],
[["hockey", :baseball], 11],
[["space", :graphics], 11],
[["medicine", :electronics], 10],
[["space", :electronics], 10],
[["electronics", :motorcycles], 10],
[["graphics", :xwindows], 10],
[["politics", :atheism], 9],
[["cryptology", :mac], 9],
[["guns", :politics], 9],
[["electronics", :graphics], 9],

[["mac", :electronics], 9],
[["cryptology", :politics], 9],
[["cryptology", :guns], 9],
[["space", :autos], 8],
[["medicine", :mac], 8],
[["cryptology", :electronics], 8],
[["autos", :motorcycles], 8],
[["guns", :religion], 8],
[["mideastpolitics", :religion], 8],
[["autos", :forsale], 7],
[["christianity", :religion], 7],
[["forsale", :electronics], 7],
[["electronics", :autos], 7],
[["mideastpolitics", :christianity], 7],
[["atheism", :motorcycles], 7],
[["space", :atheism], 7],
[["autos", :electronics], 7],
[["politics", :religion], 7],
[["politics", :space], 7],
[["medicine", :graphics], 7],
[["medicine", :christianity], 6],
[["mswindows", :xwindows], 6],
[["guns", :motorcycles], 6],
[["graphics", :autos], 6],
[["christianity", :mswindows], 6],
[["xwindows", :mac], 6],
[["mideastpolitics", :autos], 5],
[["medicine", :pc], 5],
[["cryptology", :graphics], 5],
[["graphics", :electronics], 5],
[["mideastpolitics", :guns], 5],
[["baseball", :hockey], 5],
[["mideastpolitics", :baseball], 5],
[["religion", :politics], 5],
[["medicine", :forsale], 5],
[["cryptology", :mswindows], 5],
[["medicine", :politics], 5],
[["forsale", :motorcycles], 4],
[["mideastpolitics", :motorcycles], 4],
[["pc", :forsale], 4],
[["mac", :forsale], 4],
[["mac", :autos], 4],
[["cryptology", :motorcycles], 4],

[["religion", :space], 4],
[["baseball", :forsale], 4],
[["graphics", :cryptology], 4],
[["guns", :autos], 4],
[["mswindows", :baseball], 4],
[["forsale", :mswindows], 4],
[["electronics", :cryptology], 4],
[["politics", :motorcycles], 4],
[["religion", :graphics], 4],
[["atheism", :mideastpolitics], 3],
[["medicine", :guns], 3],
[["guns", :cryptology], 3],
[["electronics", :forsale], 3],
[["medicine", :religion], 3],
[["baseball", :mswindows], 3],
[["mac", :graphics], 3],
[["mswindows", :politics], 3],
[["space", :pc], 3],
[["hockey", :motorcycles], 3],
[["space", :motorcycles], 3],
[["pc", :autos], 3],
[["hockey", :atheism], 3],
[["atheism", :guns], 3],
[["hockey", :mswindows], 2],
[["space", :christianity], 2],
[["politics", :mideastpolitics], 2],
[["atheism", :politics], 2],
[["pc", :graphics], 2],
[["space", :medicine], 2],
[["space", :mswindows], 2],
[["cryptology", :autos], 2],
[["graphics", :motorcycles], 2],
[["christianity", :motorcycles], 2],
[["politics", :cryptology], 2],
[["mswindows", :religion], 2],
[["mac", :motorcycles], 2],
[["religion", :mideastpolitics], 2],
[["mac", :baseball], 2],
[["mideastpolitics", :cryptology], 2],
[["baseball", :politics], 2],
[["medicine", :mswindows], 2],
[["baseball", :autos], 2],
[["mswindows", :christianity], 2],

[["mswindows", :space], 2],
[["motorcycles", :forsale], 2],
[["xwindows", :autos], 2],
[["xwindows", :space], 2],
[["space", :guns], 2],
[["xwindows", :baseball], 2],
[["xwindows", :forsale], 2],
[["autos", :pc], 2],
[["politics", :autos], 2],
[["graphics", :atheism], 2],
[["xwindows", :motorcycles], 2],
[["autos", :politics], 2],
[["guns", :atheism], 2],
[["medicine", :space], 2],
[["graphics", :space], 2],
[["christianity", :baseball], 1],
[["forsale", :politics], 1],
[["guns", :mideastpolitics], 1],
[["forsale", :medicine], 1],
[["forsale", :space], 1],
[["forsale", :graphics], 1],
[["guns", :mac], 1],
[["forsale", :baseball], 1],
[["guns", :christianity], 1],
[["xwindows", :politics], 1],
[["guns", :space], 1],
[["xwindows", :cryptology], 1],
[["autos", :baseball], 1],
[["autos", :graphics], 1],
[["guns", :mswindows], 1],
[["motorcycles", :baseball], 1],
[["motorcycles", :space], 1],
[["space", :mideastpolitics], 1],
[["baseball", :mac], 1],
[["baseball", :atheism], 1],
[["mac", :space], 1],
[["mac", :politics], 1],
[["baseball", :motorcycles], 1],
[["mac", :guns], 1],
[["baseball", :electronics], 1],
[["mac", :cryptology], 1],
[["baseball", :christianity], 1],
[["christianity", :xwindows], 1],

[["hockey", :electronics], 1],
[["mideastpolitics", :mswindows], 1],
[["hockey", :autos], 1],
[["mideastpolitics", :graphics], 1],
[["pc", :space], 1],
[["pc", :cryptology], 1],
[["hockey", :forsale], 1],
[["pc", :motorcycles], 1],
[["hockey", :religion], 1],
[["mideastpolitics", :forsale], 1],
[["christianity", :pc], 1],
[["christianity", :electronics], 1],
[["forsale", :hockey], 1],
[["mswindows", :atheism], 1],
[["cryptology", :atheism], 1],
[["cryptology", :xwindows], 1],
[["politics", :electronics], 1],
[["mswindows", :motorcycles], 1],
[["cryptology", :forsale], 1],
[["cryptology", :pc], 1],
[["mswindows", :hockey], 1],
[["mswindows", :autos], 1],
[["mswindows", :forsale], 1],
[["cryptology", :religion], 1],
[["cryptology", :christianity], 1],
[["mswindows", :electronics], 1],
[["politics", :mswindows], 1],
[["christianity", :space], 1],
[["graphics", :christianity], 1],
[["graphics", :baseball], 1],
[["christianity", :mac], 1],
[["space", :mac], 1],
[["graphics", :forsale], 1],
[["space", :baseball], 1],
[["electronics", :baseball], 1],
[["politics", :graphics], 1],
[["space", :religion], 1],
[["politics", :mac], 1],
[["electronics", :religion], 1],
[["electronics", :atheism], 1],
[["electronics", :medicine], 1],
[["atheism", :space], 1],
[["medicine", :mideastpolitics], 1],

[["atheism", :hockey], 1],
[["atheism", :electronics], 1],
[["atheism", :pc], 1],
[["space", :cryptology], 1],
[["atheism", :cryptology], 1],
[["atheism", :medicine], 1],
[["space", :xwindows], 1],
[["medicine", :baseball], 1],
[["religion", :motorcycles], 1],
[["religion", :autos], 1]]